**ABSTRACT**

In this technological age, we find documents are made available in digital forms which have to be classified into the topics. For solving this problem researchers focused on machine learning techniques: a general inductive process automatically builds a classifier by learning, from a set of pre classified documents, the characteristics of the categories. The main benefit of the present approach is consisting in the manual definition of a classifier by domain experts where effectiveness, less use of expert work and straightforward portability to different domains are possible[1].

**INTRODUCTION TO DATA**

Health Care Services range from basic medical diagnostics to critical emergency services. The provider follows a ticketing system for all the telephonic calls received across all the departments. Calls to the provider can be for New Appointment, Cancellation, Lab Queries, Medical Refills, Insurance Related, and General Doctor Advice etc. The Tickets have the details of Summary of the call and description of the calls written by various staff members with no standard text guidelines.

**GOAL**

Goal is to build an algorithm which can classify the calls into the sub-categories.

**DATASET DESCRIPTION**

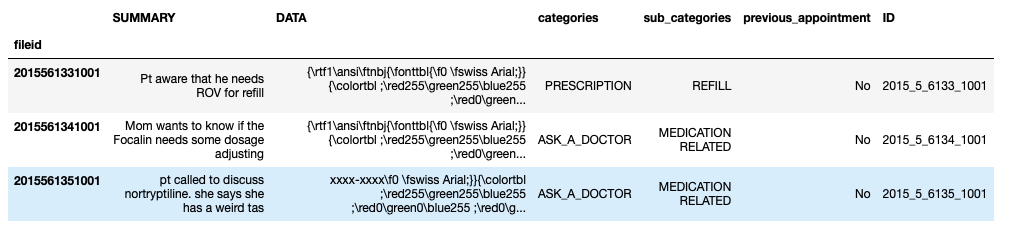
**Source:**

https://d1b10bmlvqabco.cloudfront.net/attach/iukuox3xfs6r0/ism3da5ls6r57b/ivuop6y2drl1/TextClassification\_Data.csv

**Description:**

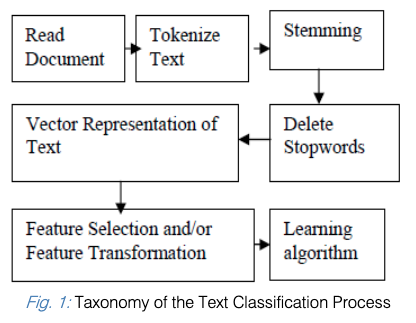
* Data is about ~84 MB.
* Health Data, notes taken from the records at the service desk
  + Each record is the conversation between a patient and the service desk employee (57,280 records)
* Features in the Data:
  + File Id, Summary, Data, Categories, Sub-Categories, Previous Appointment, ID
  + Categories: prescription, appointments, miscellaneous, ask a doctor, lab, junk
  + Sub-Categories: refill, medication related, others, sharing of health records (fax, e-mail, etc.), query on current appointment, symptoms, rescheduling, new appointment, provider, sharing of lab records (fax, e-mail, etc.), queries from pharmacy, prior authorization, lab results, cancellation, change of provider, running late to appointment, queries from insurance firm, change of hospital, follow up on previous request, change of pharmacy, junk
* DATA column is in Rich Text Format (RTF Code)[2]

**Sample Data:**



**PREPROCESSING**

Figure Source [2]

**Challenges:**

1. Data is in RTF Code format, data has to be cleaned
2. There are a lot of spelling mistakes in the data
3. Abbreviations are used in the data, ex: appt: ‘appointment’, pt: ‘Patient’
4. Thorough preprocessing of the data is required
5. Class Imbalance Problem (Fig., 2)

**Steps Followed:**

1. Understanding the data present and target classes
   * Summary and Data is to be combined to get some meaning out of the data
2. Cleaning the data
   * RTF (Rich Text Format) Code has to be cleaned and so use Regular Expression to clear RTF Code
   * Remove the Headings in RTF data like “Phone Note”, “Caller” etc., along with punctuations, numbers, special characters etc.,
   * “Junk” column is removed as it has no text in it
3. Spellings have to be checked (Optional)[3]
   * Few abbreviations are replaced by appropriate words.
4. Tokenization of Words is performed
   * **Tokenization:** The process of breaking a stream of text up into tokens that is words, phrases, symbols, or other meaningful elements is called Tokenization. The list of tokens is input to the next processing of text classification. Generally, tokenization occurs at the word level.
   * **Example:** Process of breaking stones ---> [‘process’, ‘of’, ‘breaking’, ‘stones’]
5. Stemming and Lemmatization of words is performed
   * **Stemming:** The process for decreasing deviated (or sometimes derived) words to their stem or original form (Crude method of cutting words)
     + **Example:** [‘process’, ‘of’, ‘breaking’, ‘stones’] ---> [‘process’, ‘of’, ‘break’, ‘stone’]
     + PorterStemmer is used for stemming
   * **Lemmatization:** The process of bringing all the words to base form where the base form is dictionary form
     + **Example:** [‘process’, ‘of’, ‘breaking’, ‘stones’] ---> [‘process’, ‘of’, ‘break’, ‘stone’]
     + WordNetLemmatizer is used for lemmatization
6. Stop words are removed
   * **Stop words** such as ‘and’, ‘or’, ‘is’, ‘the’, etc.. would not enhance the results rather bring down the scores, hence we are deleting the stopwords
7. Split the data into Train and Test datasets
   * sklearn’s cross\_validation is used to split the data. “stratify” argument is supplied to split into stratified samples.
8. Document term matrix is created using either TfidfVectorizer or CountVectorizer method from sklearn feature\_extraction.text package
   * Fit and transform the training data and just transform the test data
   * N-gram range is selected - Unigrams + Bigrams (1, 2)
   * Sparseness of data is determined, which would result in lesser dimension and also removes the terms which occur more frequently and less frequently
9. Training and testing of different models
10. Comparing error metrics and picking best model
11. Creating new features and train models on new data and perform testing and evaluation

**MACHINE LEARNING TECHNIQUES**

**Naive Bayes**[4]

It is just like doing a bunch of counts. If the NB conditional independence assumption actually holds, a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data. And even if the NB assumption doesn’t hold, a NB classifier still often does a great job in practice. A good bet is when we want something fast and easy that performs pretty well. Its main disadvantage is that it can’t learn interactions between features (e.g., it can’t learn that although you love movies with Brad Pitt and Tom Cruise, you hate movies where they’re together).

**Logistic Regression**[5]

Advantages of Logistic Regression are we have a lots of ways to regularize our model, and we don’t have to worry about our features being correlated, like we do in Naive Bayes. We also have a nice probabilistic interpretation, unlike decision trees or SVMs, and we can easily update our model to take in new data (using an online gradient descent method), again unlike decision trees or SVMs. We can use it if we want a probabilistic framework (e.g., to easily adjust classification thresholds, to say when we are unsure, or to get confidence intervals) or if we expect to receive more training data in the future that we want to be able to quickly incorporate into your model.

**Random Forest**[6]

Random Forest trains each tree independently, using a random sample of the data. This randomness helps to make the model more robust than a single decision tree, and less likely to overfit on the training data.

**Gradient Boosting**[7]

GBT build trees one at a time, where each new tree helps to correct errors made by previously trained tree.

**Evaluation Metrics**

There are various methods to determine effectiveness; however, precision, recall, and accuracy are most often used. To determine these, one must first begin by understanding if the classification of a document was a true positive (TP), false positive (FP), true negative (TN), or false negative (FN).

Kappa Metric[8], ROC curves can also be used in case of imbalanced classes.

**RESULTS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model Name** | **Optimum Parameters** | **Accuracy** | **Precision** | **Recall** | **Kappa Score** | **Time Taken to build** |
| Naïve Bayes | Default | 0.625684014 | 0.50534707 | 0.358578716 | 0.561746242 | 621s |
| Logistic Regression | C: 10 | 0.706426825 | 0.549921598 | 0.60356259 | 0.667943189 | 28.3s |
| Random Forest | Max\_depth:” 500, max\_feature s: auto, criterion:  gini | 0.704214693 | 0.60318234 | 0.480315247 | 0.65723304 | 1min 7s |
| Gradient Boosting | Default | 0.716614274 | 0.586582074 | 0.532795069 | 0.674009701 | 29 min 1s |
| Logistic Regression with added Features | C: 10 | 0.67341949 | 0.533516799 | 0.530343638 | 0.626139995 | 3 min 8s |
| Gradient Boosting with added Features | Default | 0.72179532 | 0.579006109 | 0.527740469 | 0.679159976 | 32 min 14s |

**BEST ALGORITHM**

Except for Naive Bayes all models have got more than 70% in Accuracy, Precision, Recall, F1 and more than 60% Kappa score. But, Logistic regression performed better than others in categories like “cancellation” and “symptoms”, but not great in classifying the “medication related”, “lab results” and “new appointment” though they are larger in number.

**Insights from Confusion Matrix:**

* Naive Bayes and Random forest failed to predict at least one document as ​ ‘change of hospital’​ and​ ‘change of pharmacy’​
* Logistic Regression classified some of above categories. In addition to that it rightly classified categories like ‘symptoms’​ and ‘​cancellation’​ in high number
* Logistic regression failed to classify most of ​medication related​, most of them are misclassified as ‘symptoms’ and ‘​refill’
* Most misclassified classes are,
  + Cancellation vs rescheduling
  + Medication related vs (others, refill, symptoms)
  + New appointment vs (medication related, others, rescheduling, queries on current appointment
  + Others vs (medication related, new appointment, sharing of health records)
  + Refill vs medication related
  + Sharing of lab records vs lab results
  + Symptoms vs medication related

**FEATURE ENGINEERING**

* Combine certain classes which are similar
  + Sharing of health records and Sharing of lab records are similar and are talking about same thing like patient requesting for records, or hospital management trying to send records. So, these two are combined to “sharing of records”.
* Engineer new features, terms common for a category and uncommon across corpus
  + In general, tf-idf gives the importance of terms in a document. So, to get the most uncommon words for a category, all the documents are grouped by categories. Finally there are 20 documents. Now, tf-idf is calculated and word cloud is built for the words thus obtained, which in return shows the important terms for a category that are not present in any other categories.
  + For each word a column is generated and is attached with original train data. The column values are binary, 0 means word is not present and 1 means word is present. These new features are added to original train matrix, and models are trained.

**FINAL RESULTS**

Best results were obtained with Gradient Boosting. For confusion matrix of Gradient Boosting refer figure 4.

**PLOTS**

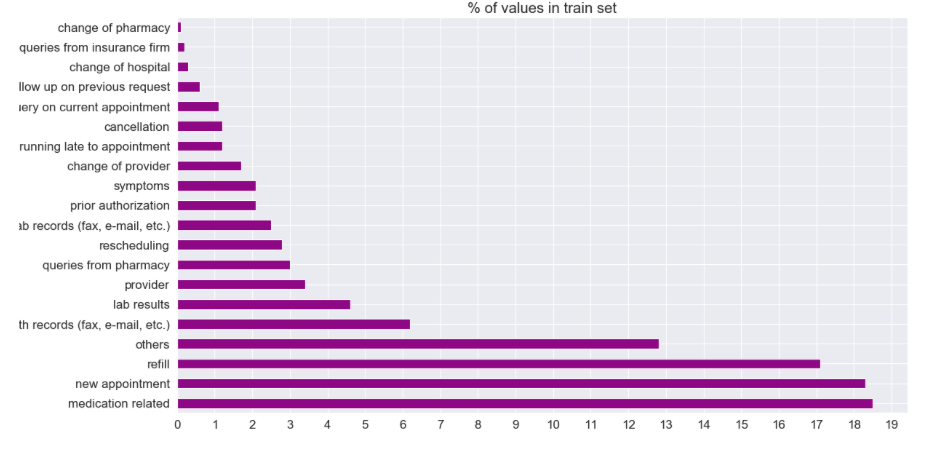
****

Fig 2: Class Imbalance Reference

****

Fig 3: Word Cloud for each Category

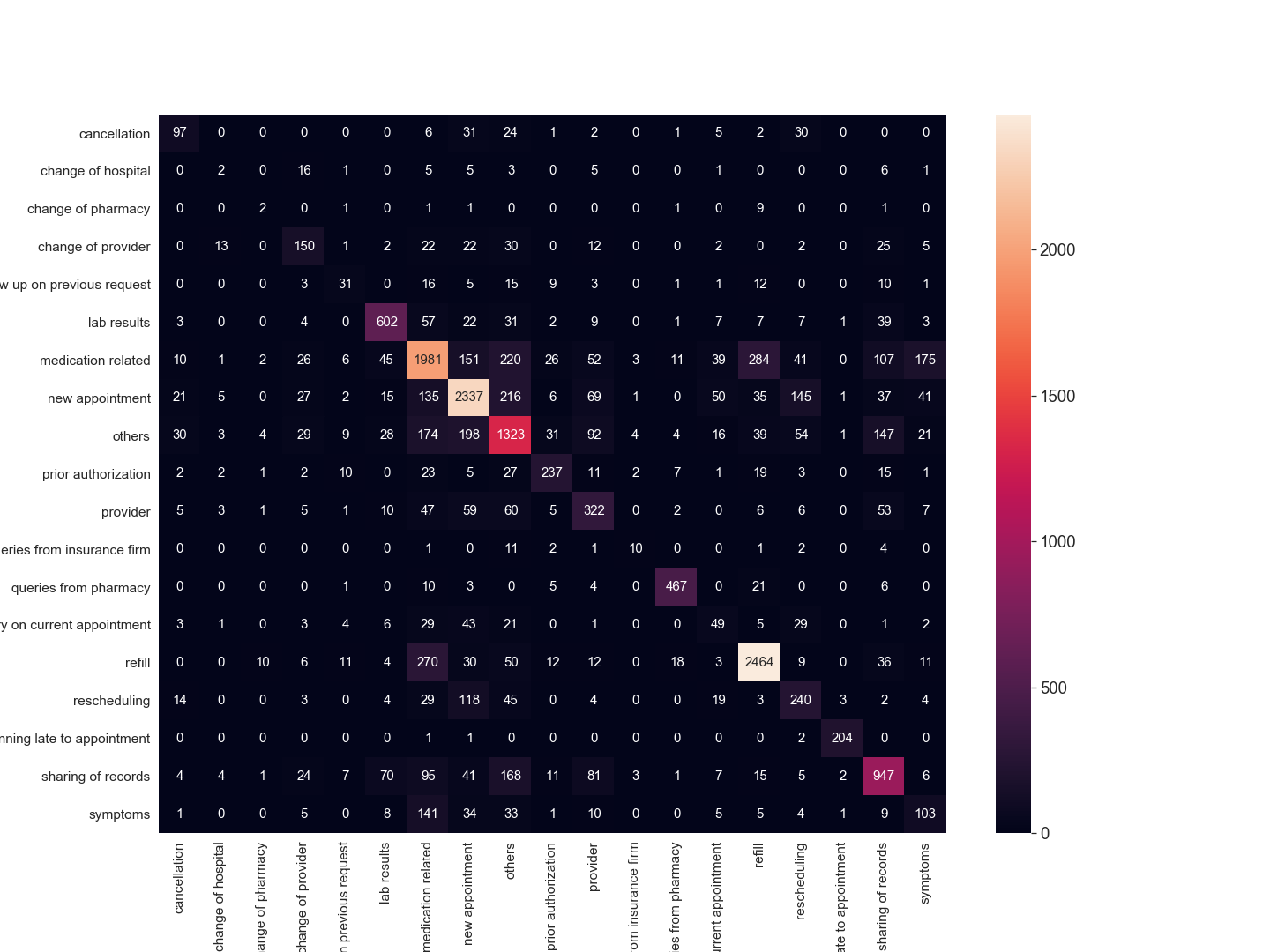


Fig 4: Confusion Matrix

**REFERENCES**

1. Bhavani Dasari, Dr. Venu Gopala Rao. K, Durga. " Text Categorization and Machine Learning Methods: Current State Of The Art." Global Journal of Computer Science and Technology[Online], (2012): n. pag. Web. 21 May. 2019
2. Rich Text Format https://en.wikipedia.org/wiki/Rich\_Text\_Format Retrieved on 2019 - 05 - 21
3. Peter Norvig, “How to Write a Spelling Corrector” found at https://norvig.com/spell-correct.html Retrieved on 2019 - 05 - 21
4. Maron, M. E. (1961). ["Automatic Indexing: An Experimental Inquiry"](http://delivery.acm.org/10.1145/330000/321084/p404-maron.pdf?ip=146.244.117.157&id=321084&acc=ACTIVE%20SERVICE&key=F26C2ADAC1542D74%2E144ADCEE99EDDC29%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1551394905_96b60790765feedd11602dd1dfde40dd) (PDF). [Journal of the ACM](https://en.wikipedia.org/wiki/Journal_of_the_ACM). 8 (3): 404–417. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):10.1145/321075.321084.
5. Walker, Strother H., and David B. Duncan. “Estimation of the Probability of an Event as a Function of Several Independent Variables.” Biometrika, vol. 54, no. 1/2, 1967, pp. 167–179. JSTOR, www.jstor.org/stable/2333860.
6. Ho, Tin Kam (1995). [Random Decision Forests](https://web.archive.org/web/20160417030218/http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf) (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282. Retrieved on 2019 - 05 - 21.
7. Friedman, J. H. (February 1999). "Greedy Function Approximation: A Gradient Boosting Machine" (PDF).
8. McHugh, Mary L. “Interrater reliability: the kappa statistic.” Biochemia medica vol. 22,3 (): 276-82.